var mysql = require("mysql");

var pool = mysql.createPool({

host : "localhost",

database : "6666",

user : "root",

password : ""

});

function getConnection()

{

return new Promise(function(resolve, reject){

pool.getConnection((err, conn)=> {

if(err){

reject(err);

}

else{

resolve(conn);

}

});

});

}

function executeQuery(conn, query,params){

return new Promise((resolve, reject)=>{

conn.query(query, params, (err, result)=>{

if(err) {

reject(err);

}

else{

resolve(result);

}

})

});

}

module.exports = {

getConnection:getConnection,

executeQuery:executeQuery,

}

const express = require("express");

const app = express();

var db = require("./database");

// yang disorot diminggu ini json web token

const jwt = require("jsonwebtoken");

app.use(express.urlencoded({extended:true}));

app.use(express.json());

app.get("/insert", async (req, res)=>{

let conn = await db.getConnection();

try{

let hasil = await db.executeQuery(conn, "INSERT INTO USER VALUES ('?','?','?')",[1,1,1]);

res.send(hasil);

}

catch(ex){

res.send(ex);

}

})

//{ nama, password, nomorhp, alamat, status(0 untuk pemilik 1 untuk peminjam), saldo }

app.post("/api/register", async (req, res)=>{

let conn = await db.getConnection();

let nomorhp = req.body.nomorhp;

let nama = req.body.nama;

let password = req.body.password;

let alamat = req.body.alamat;

let status = parseInt(req.body.status);

let saldo = parseInt(req.body.saldo)

if(nomorhp.length == 0 || nama.length == 0 || password.length == 0 || alamat.length == 0 || String(status).length == 0 || String(saldo).length == 0){

return res.send("Tidak boleh ada field yang kosong")

}

if(status == 1 || status == 0)

{

try{

let result = await db.executeQuery(conn, "INSERT INTO USER VALUES(?,?,?,?,?,?)", [nomorhp, nama, password, alamat, status, saldo])

res.send({nomorhp:nomorhp, password:password})

}

catch(ex){

if(ex.code == "ER\_DUP\_ENTRY"){return res.send({message : "Nomor sudah terpakai"})}

else if(ex.code == "ER\_BAD\_FIELD\_ERROR"){return res.send({message : "Semua Field(s) harus diisi"})}

res.send(ex);

}

}

else{

return res.send({message : "Status hanya berisi 1 dan 0"})

}

})

app.post("/api/login", async (req, res)=>{

let conn = await db.getConnection();

let nomorhp = req.body.nomorhp;

let password = req.body.password;

try{

let cari = await db.executeQuery(conn, "SELECT \* FROM USER WHERE NOMORHP = ? AND PASSWORD = ?", [nomorhp, password]);

if(cari.length > 0)

{

const token = jwt.sign({

nomorhp : nomorhp,

tipe : cari[0].status

}, "tugas6");

res.status(200).send(token);

}

else{

res.send("Username atau password salah!")

}

}

catch(ex){

res.send(ex);

}

})

app.post("/api/topup", async (req, res)=>{

let conn = await db.getConnection();

let nomorhp = req.body.nomorhp;

let saldo = parseInt(req.body.saldo);

const token = req.header("x-auth-token");

if(!token){

return res.status(401).send("Token not found")

}

let user = {};

try {

user = jwt.verify(token, "tugas6");

} catch (error) {

return res.status(400).send("Invalid Token")

}

if((new Date().getTime()/1000) - user.iat > 3600){

return res.status(400).send("token expired");

}

if(user.tipe == 1){

try{

let cari = await db.executeQuery(conn, "SELECT \* FROM USER WHERE NOMORHP = ?", [nomorhp]);

if(cari.length > 0){

let update = await db.executeQuery(conn, "UPDATE USER set saldo = ? WHERE NOMORHP = ?", [(cari[0].saldo+ saldo), nomorhp])

res.send({nomorhp : nomorhp, saldo : cari[0].saldo + saldo})

}

else{

res.send("Nomor hp tidak terdaftar")

}

}

catch(ex){

res.send(ex);

}

}

else{

res.send("Topup hanya untuk peminjam");

}

})

app.post("/api/house", async (req, res)=>{

let conn = await db.getConnection();

let nama = String(req.body.nama);

let alamat = req.body.alamat;

let tipe = req.body.tipe;

let max = req.body.max;

let harga = parseInt(req.body.harga);

kode = nama.substr(0,2).toUpperCase()

let autogen = await db.executeQuery(conn, "SELECT IFNULL(max(substr(id, 3, 3)),0) + 1 AS GEN FROM HOUSE WHERE SUBSTR(id,1,2) = ?", [kode])

let digit = autogen[0].GEN + ""

digit = digit.padStart(3, "0")

digit = kode + digit

const token = req.header("x-auth-token");

if(!token){

return res.status(401).send("Token not found")

}

let user = {};

try {

user = jwt.verify(token, "tugas6");

} catch (error) {

return res.status(400).send("Invalid Token")

}

if((new Date().getTime()/1000) - user.iat > 3600)

{

return res.status(400).send("token expired");

}

if(user.tipe == 0)

{

try{

let cari = await db.executeQuery(conn, "INSERT INTO HOUSE VALUES(?,?,?,?,?,?)", [digit, nama, alamat, tipe, max, harga]);

res.send({kodeRumah : digit, ...req.body})

}

catch(ex){

if(ex.code == "ER\_BAD\_FIELD\_ERROR"){return res.send({message : "Semua field(s) harus diisi"})}

res.send(ex);

}

}

else{

return res.send("Anda tidak memiliki autorization! bukan pemilik!")

}

})

app.put("/api/house/:id", async (req, res)=>{

let conn = await db.getConnection();

let id = req.params.id;

let nama = req.body.nama;

let alamat = req.body.alamat;

if (nama.length == 0 || alamat.length == 0){

return res.send("Tidak boleh ada field(s) yang kosong!");

}

const token = req.header("x-auth-token");

if(!token){

return res.status(401).send("Token not found")

}

let user = {};

try {

user = jwt.verify(token, "tugas6");

} catch (error) {

return res.status(400).send("Invalid Token")

}

if((new Date().getTime()/1000) - user.iat > 3600){

return res.status(400).send("token expired");

}

if(user.tipe == 0){

try{

let cari = await db.executeQuery(conn, "UPDATE HOUSE SET NAMA = ?, ALAMAT = ? WHERE ID = ?", [nama, alamat, id]);

if(cari.affectedRows == 1){

let result = await db.executeQuery(conn, "SELECT \* FROM HOUSE WHERE ID = ?",[id])

res.send(result[0]);

}

else{

res.send("ID rumah tidak ditemukan!")

}

}

catch(ex){

if(ex.code == "ER\_BAD\_FIELD\_ERROR"){return res.send({message : "Semua field(s) harus diisi"})}

res.send(ex);

}

}

else{

return res.send("Anda tidak memiliki autorization! bukan pemilik!")

}

})

app.post("/api/rent", async (req, res)=>{

let conn = await db.getConnection();

let nohp\_peminjam = req.body.nohp\_peminjam;

let nohp\_pemilik = req.body.nohp\_pemilik

let id = req.body.id;

let startdate = new Date(req.body.startdate);

let enddate = new Date(req.body.enddate);

const token = req.header("x-auth-token");

if(!token){

return res.status(401).send("Token not found")

}

let user = {};

try {

user = jwt.verify(token, "tugas6");

} catch (error) {

return res.status(400).send("Invalid Token")

}

if((new Date().getTime()/1000) - user.iat > 3600){

return res.status(400).send("token expired");

}

if(user.tipe == 1){

try{

let saldo = await db.executeQuery(conn, "SELECT \* FROM USER WHERE NOMORHP = ?",[nohp\_peminjam]);

saldo = saldo[0].saldo;

let harga = await db.executeQuery(conn, "SELECT \* FROM HOUSE WHERE ID = ?", [id])

harga = harga[0].harga;

let total = parseInt(req.body.total)

if(saldo - total < 0){

return res.send("Saldo tidak cukup!");

}

let tambahsaldopemilik = await db.executeQuery(conn, "UPDATE USER SET SALDO = SALDO + ? WHERE NOMORHP = ?",[total, nohp\_pemilik])

let update = await db.executeQuery(conn, "UPDATE USER SET SALDO = ? WHERE NOMORHP = ?",[(saldo - total), nohp\_peminjam])

let cari = await db.executeQuery(conn, "INSERT INTO SEWA VALUES(?,?,?,?,?,?)", ['', nohp\_peminjam, nohp\_pemilik, id, startdate, enddate]);

res.send({idsewa : cari.insertId, ...req.body})

}

catch(ex){

if(ex.code.includes("ER\_BAD")){return res.send({message : "Semua field(s) harus diisi"})}

else if(ex.sqlMessage.includes("fk\_nohp\_pemilik")){return res.send({message:"nomor hp pemilik tidak ditemukan!"})}

else if(ex.sqlMessage.includes("fk\_rumah")){return res.send({message:"id rumah tidak ditemukan!"})}

else if(ex.sqlMessage.includes("fk\_nohp\_peminjam")){return res.send({message:"nomor hp peminjam tidak ditemukan!"})}

res.send(ex);

}

}

else{

return res.send("Anda tidak memiliki autorization! bukan peminjam!")

}

})

app.get("/api/history/:nomorhp", async (req, res)=>{

let conn = await db.getConnection();

let nomorhp = req.params.nomorhp;

const token = req.header("x-auth-token");

if(!token){

return res.status(401).send("Token not found")

}

let user = {};

try {

user = jwt.verify(token, "tugas6");

} catch (error) {

return res.status(400).send("Invalid Token")

}

if((new Date().getTime()/1000) - user.iat > 3600){

return res.status(400).send("token expired");

}

if(user.tipe == 1){

try{

let cari = await db.executeQuery(conn, "SELECT \* FROM SEWA WHERE NOHP\_PEMINJAM = ?", [nomorhp]);

if(cari.length > 0){

res.send(cari);

}

else{

res.send("Hasil tidak ditemukan!")

}

}

catch(ex){

if(ex.code.includes("ER\_BAD")){return res.send({message : "Semua field(s) harus diisi"})}

res.send(ex);

}

}

else{

return res.send("Anda tidak memiliki autorization! bukan peminjam!")

}

})

app.get("/api/search", async (req, res)=>{

let conn = await db.getConnection();

let query = req.query

let sql = "SELECT \* FROM HOUSE WHERE 1 = 1";

let startdate, enddate;

if(query.startdate === undefined || query.enddate === undefined){

return res.send({message : "startdate dan enddate harus diisi"})

}

if(query.startdate.length == 0 || query.enddate.length == 0){

return res.send({message:"stardate dan enddate tidak boleh kosong!"})

}

startdate = query.startdate

enddate = query.enddate

let booking = `SELECT \* FROM sewa WHERE startdate >= '${startdate}' AND enddate <= '${enddate}' group by id\_rumah`

if(query.nama !== undefined){

if(query.nama.length == 0){

return res.send({message : "Field nama tidak boleh kosong!"})

}

sql += ` AND nama like '${query.nama}%'`

}

if(query.tipe !== undefined){

if(query.tipe.length == 0){

return res.send({message : "Field tipe tidak boleh kosong!"})

}

sql += ` AND tipe = '${query.tipe}'`

}

if(query.guest !== undefined ){

if(query.guest.length == 0){

return res.send({message : "Field guest tidak nama kosong!"})

}

sql += ` AND max <= ${query.guest}`

}

let caritidak = await db.executeQuery(conn, booking,[]);

caritidak.forEach(element => {

sql+= ` AND ID <> '${element.id\_rumah}'`

});

const token = req.header("x-auth-token");

if(!token){

return res.status(401).send("Token not found")

}

let user = {};

try {

user = jwt.verify(token, "tugas6");

} catch (error) {

return res.status(400).send("Invalid Token")

}

if((new Date().getTime()/1000) - user.iat > 3600){

return res.status(400).send("token expired");

}

if(user.tipe == 1){

try{

let cari = await db.executeQuery(conn, sql, []);

if(cari.length > 0){

res.send(cari);

}

else{

res.send("Hasil tidak ditemukan!")

}

}

catch(ex){

res.send(ex);

}

}

else{

return res.send("Anda tidak memiliki autorization! bukan peminjam!")

}

})

app.listen(3000, function(){

console.log("Listening to port 3000");

})